How to Double Your Graphics Performance Without New Hardware

Harald Smit (NVIDIA), Allen Jensen (NVIDIA), Avi Shapira (Graphic Remedy)
Agenda

• Goals
• Driver Performance Tuning
• Limits to Graphics Performance
• Benchmarks
• Techniques
• Tools
  • Demo of gDEBugger
• Q & A
Goals

• Improve Overall Performance
  • Deliver a solution!

• Communication
  • Tune for both today and tomorrow
  • Ask questions - interactive discussion

• Cooperation
  • Performance tuning is often a balance between driver and application

• Learn New Techniques
  • Best Practices, Tools
• Performance tuning isn’t easy
  • Problem continually changes
    • New GPU’s, CPU’s, OS’es, drivers, applications
  • Many different configurations
    • Low end, high end, mixing Compute / Graphics

• Today’s talk is
  • OpenGL focused
  • Targeted for Quadro / Professional app’s
  • Windows perspective, but same for Linux
Driver Performance Tuning

• Why can’t all performance problems be solved in the driver?
  • OS, application, GPU
• GPU bottleneck can be a good thing
• Problem areas for Driver Developers
  • Applications which block debuggers/tools
  • Applications which change behavior based on load
Performance within NVIDIA

• Delivering top performance is key
  • Maintaining high quality is required
• Automated Test Labs
  • Track performance / quality regressions
• Performance Labs
  • Runs the latest and greatest
• Driver Development
  • Teams focused per API / business area
Limits to Graphics Performance

• Data Movement
• Methods of Data Movement
• State changes
• Application overheads
• Memory usage
• Synchronization
• Examples
Data Movement

• Just touching the data is often most of the performance cost
  – VBO/Display list good way to improve!
  – Locality of data
  – Try not to touch the data per frame
    • Example: copying buffers between threads
Methods of Data Movement

• How is the vertex data sent?
  – Begin/End
  – Vertex Array
  – VBO
  – Display List
Methods of Data Movement

• Worst
  – Begin/End
    • Easy to use but bad performance
    • Spoon feeding
    • Multiple layers per call
    • Only a few words of data per API call
    • Function calls are expensive
Methods of Data Movement

• Better
  – Vertex arrays
    • Data still needs moving
    • More data sent with each call
    • Many words per API call
Methods of Data Movement

• Best!
  – VBO
    • On card data can be referenced
    • Easily editable
      – But static for best performance!
  • Vertex data only
  • White paper from SPEC
Methods of Data Movement

• Best!
  – Display List
    • Best opportunity for graphics optimization
    • Contains vertex data and state changes
    • Not perfect though
      – Not editable
      – State input/output
      – Data de-referenced at compile time
      – Memory footprint (working on an extension!)
Application Overheads

- Performance not always a driver problem
- As GPU capabilities increase:
  - Often faster to just send everything
    - No level of detail
    - No bounding boxes
    - Uniform data
- Adaptive workloads
  - tend to slow mid range to high end systems down
  - hard to tune
Memory Usage

• Address space
  – Out of memory issues - app, driver, OS
  – Serious issue in 32 bit space
  – Both performance and quality issue

• Go to 64 bit or..
  – Try the /3GB flag
  – Compile with LARGEADDRESSAWARE
Synchronization

• While required in many cases, use only when needed
• The following functions may cause serious latency / performance issues
  – glFlush
  – glFinish
  – glMakeCurrent
  – glGet
  – glGetError
  – glReadPixels
State Changes

• Hardware must be reprogrammed to handle different user state
• The overhead of these changes can kill graphics performance
• Try to group primitives with the same state
• Minimize redundant operations
Examples

Edged polygons

```c
    glEnable(light)
    glEnable(texture)
    // draw prim
    glBegin/glVertex/glVertex/glVertex/glVertex/glEnd
    glDisable(texture)
    glDisable(light)
    glEnable(polygonOffset)
    // drawEdge
    glBegin/glVertex/glVertex/glVertex/glVertex/glEnd
    glDisable(polygonOffset)
```

- Problem: too much time setting up hardware
- Solution: group edges/prims
Examples

Stippled lines

```gl
    glEnable(stipple)
    glBegin/glEnd
    glDisable(stipple)
    glBegin/glEnd
    glEnable(stipple)
    glBegin/glEnd
    glDisable(stipple)
```

- Problem: state changes expensive
- Solution: grouping stippled and non-stippled lines
Examples

Vertex array setup per object

```c
setEnabledClientState(GL_VERTEX)
setEnabledClientState(GL_NORMAL)
VertexBufferPointer(value)
NormalPointer(value)
DrawElements
NormalPointer(NULL)
VertexBufferPointer(NULL)   ?? - seen it!
DisableClientState(GL_VERTEX)
DisableClientState(GL_NORMAL)
```

- Problem: array setup dirtied too often
- Solution: track array state in application
Examples

Using driver for redundant state management

```java
if (!isEnabled(GL_LIGHTING))
    glEnable(GL_LIGHTING)
```

- Problem: Driver needs to switch gears to respond to isEnabled (threading issues etc)
- Solution: Application flag..

```java
if (!app.glLighting) {
    glEnable(GL_LIGHTING)
    app.glLighting = TRUE;
}
```
Examples

Sending non-uniform state within Begin/End

```plaintext
glBegin
glColor
 glVertex
 glVertex
 glVertex
 glNormal
 glVertex
 glVertex
 glEnd
```

- **Problem**: Driver often builds constant data structures for optimal hardware performance
- **Solution**: Send per prim attributes outside Begin/End. Don't go to the trouble of removing redundant values
Windows Vista vs. XP

• OS plays a big role in performance
• Windows XP
  • Mature, tuned for years
• Windows Vista
  • GDI / OpenGL interaction
  • Avoid front buffer rendering

• See the whitepaper here:
  • http://www.spec.org/gwpg/publish/vista_paper.html

• TDR’s (Timeout Detection & Recovery)
  • http://www.microsoft.com/whdc/device/display/wddm_timeout.mspx
Benchmarks

- **SPEC GWPG** ([http://www.spec.org/gwpg/](http://www.spec.org/gwpg/))
  - **Viewperf** (current 10.0)
    - Replay traces of models from many different application
    - Highlights GPU / driver performance
    - Now being used for Energy Star power measurement

- **SPEC APC**
  - Application scripted performance tests
    - Highlights full solution performance
    - 3dsmax, Maya, ProE, SolidEdge, Solidworks, UGS NX
Benchmarks (cont.)

- Remember to turn off sync to Vblank!
- More benchmarks are always needed
  - Easier for driver team to focus on important areas of an application
  - More industry recognition (free advertising)
- What about your app / customer?
  - APC test or Viewperf viewset?
  - Please see Allen after the class to pursue
Additional Performance Topics

• ACE (Application Configuration Engine)
• Multi-threading
• Multi-GPU
• NVSG
• Performance drivers
• Extensions
ACE

• Application Configuration Engine
  • Automatic application profile selection
• Powerful performance tool
  • Allows directed optimizations
• What we need
  • Application executable name / path
  • Profile name
  • Desired GUI profile settings
Multi-threading

- Leverage today’s CPU’s architectures
- Drive GPU harder
- Driver model fully supports this at high performance
- Separate tasks
- Reduce intra-thread communication
Other Performance Topics

- **Multi GPU**
  - **SLI**
    - Perf tuning is up to us
    - FSAA can perform very well here
    - *Mosaic Mode (QuadroPlex only) - great new feature*
      - [http://www.nvidia.com/object/quadro_sli_mosaic_mode.html](http://www.nvidia.com/object/quadro_sli_mosaic_mode.html)

- **Compute**
  - Beyond the scope of this talk, but a powerful new option in combination with graphics

- **GPU affinity**
  - If you know how you want to balance your GPU workload
Other Performance ... (cont.)

- NVSG
  - Scene graph
- Performance drivers
- Extensions
- What about your application?
Tools

• General CPU Performance tuning
  • VTune (Intel)
  • Code Analyst (AMD)

• GPU Performance tuning
  • NVIDIA Perfkit
  • GLExpert
  • gDEBugger
Q&A

• ?
Summary

• What we all can do:
  • Keep communication channels open
  • Work together on optimizations
  • Collaborate on benchmarks

• What you can do:
  • Review your code
    • Look for pitfalls noted
    • Change to best practices - VBO’s, Display List
  • Try the tools like gDEBugger
  • Send us more benchmarks / tests